

HOW TO TAME YOUR SPARSITY CONSTRAINTS

J. A. LOPEZ

ABSTRACT. We show that designing sparse H_∞ controllers, in a discrete (LTI) setting, is easy when the controller is assumed to be an FIR filter. In this case, the problem reduces to a static output feedback problem with equality constraints. We show how to obtain an initial guess, for the controller, and then provide a simple algorithm that alternates between two (convex) feasibility programs until converging, when the problem is feasible, to a suboptimal H_∞ controller that is automatically stable. As FIR filters contain the information of their impulse response in their coefficients, it is easy to see that our results provide a path of least resistance to designing sparse robust controllers for continuous-time plants, via system identification methods.

1. INTRODUCTION

Recently there has been much interest in designing controllers that satisfy so-called “sparsity constraints” – a catch-all term for constraints that arise from information flow restrictions [20, 28, 15, 27], by the control-loop topology [24, 9], etc. The main difficulty in sparse controller design comes from the obscure relationship between sparsity constraints, imposed on the pattern of the closed-loop transfer matrix, and the closed-loop state-space description; which is where we prefer to work since modern control theory relies heavily on tools from linear algebra. As with non-sparse robust control design, there seem to be two principal roads to controller synthesis: the Youla parametrization and the linear matrix inequality (LMI) solutions. Some sparse control researchers have focused on determining theoretical conditions under which the Youla parametrization solution remains convex under sparsity constraints [20]; while others have made progress by adapting existing LMI tools so that they can extract sparse controllers by assuming an easier-to-work-with structure on the controller or in the LMI formulation [9, 19, 17]. Our approach also imposes a structure on the problem: we assume the controller is an FIR filter (and therefore also assume the plant is strongly stabilizable) but we do not require the sparsity pattern to be in block form nor do we impose a structure on the LMI factors. Moreover, our method uses an alternating scheme that culminates with a controller and a certificate of stability and robust performance. Perhaps the closest work to ours is [14], where the authors also use an alternating scheme to design sparse controllers for the H_2 (static) state-feedback case. Our approach can be considered a generalization of their work since it includes static state-feedback controller design as a special case and can easily incorporate H_2 constraints, if desired.

Date: May 30, 2015.

Key words and phrases. robust control, sparse controller.

The author is with the Department of Electrical & Computer Engineering, Northeastern University, MA 02115, USA. E-mail: lopez.jo@husky.neu.edu. This work was supported in part by the NSF Integrative Graduate Education and Research Traineeship (IGERT) program under grant DGE-0654176.

However, unlike [14], we will not search for an optimal solution since there is little to gain from achieving the lowest H_∞ norm [23].

There are many advantages to using an FIR controller. We will show that by imposing this form on the controller, the problem of synthesizing (sparse) suboptimal H_∞ controllers reduces to a static output-feedback problem, for an augmented system, with the sparsity constraints entering the problem through simple equality constraints. Although imposing an FIR structure also assumes the plant is strongly stabilizable, this assumption is not overly restrictive since it is not practical to implement unstable controllers, anyway [5, 18, 6]. Apart from this assumption, we only assume that the plant is detectable and that the control inputs do not directly feed-through to the output. Thus, our method only has mild restrictions and we will demonstrate that the resulting controllers attain excellent bounds on the H_∞ norm. The controllers are ultimately obtained using a simple algorithm that alternates between two (convex) feasibility programs until convergence, when the problem is feasible – while we do not pursue convergence guarantees, we take comfort in knowing that our algorithm is simpler than other alternating schemes (ADMM and expectation-maximization) for which convergence proofs are available [2, 1]. Furthermore, other alternating schemes like “D-K iteration” have had success in the past.

In the following sections we will state the problem, present our main contribution, and demonstrate its use with examples.

1.1. Preliminaries. Our notation is standard from the robust control literature [23, 29, 11]. We consider the standard robust control loop shown in Figure 1: where G represents the discrete-time plant, w represents the disturbance inputs, u the control inputs, z the regulated outputs, and y the outputs available to the discrete-time controller, K [23].

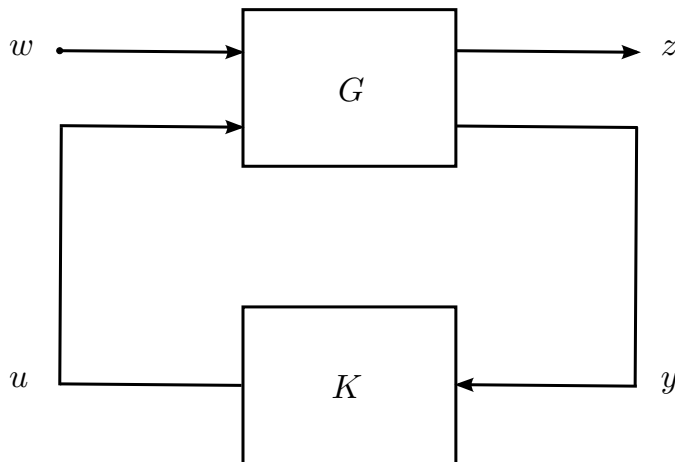


FIGURE 1. Closed-loop system

The state-space description is given by:

$$(1) \quad \begin{aligned} x_{k+1} &= Ax_k + B_1 w_k + B_2 u_k \\ z_k &= C_1 x_k + D_{11} w_k + D_{12} u_k \\ y_k &= C_2 x_k + D_{21} w_k \end{aligned}$$

For this system, we would like to minimize the \mathcal{H}_∞ norm of the transfer function from the disturbance inputs to the regulated outputs:

$$(2) \quad T_{zw} = \left[\begin{array}{c|c} A_{CL} & B_{CL} \\ \hline C_{CL} & D_{CL} \end{array} \right]$$

where $(A_{CL}, B_{CL}, C_{CL}, D_{CL})$ is the system with an output feedback controller $u = Ky$ in the loop. Formally, we consider Problem 1 below.

Problem 1. Given $\mu > 0$ and a binary pattern matrix, S , find a stabilizing controller $K \in S$ such that $\|T_{zw}\|_\infty^2 < \mu$.

Assumptions: The assumptions we make are that the pair (A, B_2) is strongly stabilizable, (A, C_2) is detectable, and that $D_{22} = 0$ [12].

For our solution, we turn to a standard result in the robust control literature [10, 3, 12, 11].

Lemma 1. *The inequality $\|T_{zw}\|_\infty^2 < \mu$ holds if, and only if, there is a symmetric matrix P such that*

$$(3) \quad \begin{bmatrix} P & AP & B & 0 \\ PA^T & P & 0 & PC^T \\ B^T & 0 & I & D^T \\ 0 & CP & D & \mu I \end{bmatrix} > 0$$

is feasible.

Next, we include a review on FIR realizations from [22, 21]. For this review, we will temporarily abuse the notation by also using z for the traditional z -transform variable. If $Q(z)$ is an, l -input, FIR filter with $Q(z) = \sum_{i=0}^{n-1} Q_i z^{-i}$, then one state-space realization is:

$$(4) \quad Q = \left[\begin{array}{c|c} A_q^{n-1} & E_{n-1}^{n-1} \\ \hline [Q_{n-1}, \dots, Q_1] & Q_0 \end{array} \right]$$

with

$$(5) \quad A_q^n = \begin{bmatrix} 0 & I_l & 0 & \dots & 0 \\ 0 & 0 & I_l & \dots & 0 \\ \vdots & \vdots & \vdots & \dots & I_l \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix} \in \mathbb{R}^{nl \times nl}$$

$$(6) \quad E_1^n = \begin{bmatrix} I_l \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \dots, E_n^n = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ I_l \end{bmatrix} \in \mathbb{R}^{nl \times l}$$

For example, if

$$(7) \quad Q(z) = [1 + 2z^{-1} + 3z^{-2} \quad 1 + 2z^{-1} + 3z^{-2} \quad 1 + 2z^{-1} + 3z^{-2}]$$

then we will assume its state-space description has the form:

$$(8) \quad A_q = \begin{bmatrix} 0 & I_3 \\ 0 & 0 \end{bmatrix}, B_q = \begin{bmatrix} 0 \\ I_3 \end{bmatrix}, C_q = [Q_2 \quad Q_1], D_q = Q_0, \text{ with} \\ Q_0 = [1 \quad 1 \quad 1], Q_1 = [2 \quad 2 \quad 2], \text{ and } Q_2 = [3 \quad 3 \quad 3].$$

Thus, besides being automatically stable, FIR filters have the satisfying property of clearly identifying entries in its transfer function with entries in its state-space description. (Clearly, $Q(j, k) = 0$ if, and only if, $Q_i(j, k) = 0$ for all i .) This obviates any difficulty that comes with trying to associate entries in a general transfer matrix (i.e., one not required to be an FIR filter) with its state-space description via the transformation $G(z) = C(zI - A)^{-1}B + D$ which has an unfriendly inversion in the way. Furthermore, for a fixed n , A_q and B_q are also fixed. We will use this property, in the next section, where we show that Problem 1 reduces to a static output feedback problem, for an augmented system, when we require that:

$$(9) \quad K = \left[\begin{array}{c|c} A_q & B_q \\ \hline C_q & D_q \end{array} \right].$$

2. MAIN RESULT

In this section, we prove the following claim.

Lemma 2. *Problem 1 reduces to a static output feedback problem for the augmented system (A_o, B_o, C_o, D_o) where*

$$(10) \quad A_o = \begin{bmatrix} A & 0 \\ B_q C_2 & A_q \end{bmatrix}, B_o = \begin{bmatrix} B_1 \\ B_q D_{21} \end{bmatrix}, \\ C_o = [C_1 \quad 0], D_o = D_{11}.$$

Proof. With K as in (9), the closed-loop system is given by

$$(11) \quad A_{CL} = \begin{bmatrix} A + B_2 D_q C_2 & B_2 C_q \\ B_q C_2 & A_q \end{bmatrix}, B_{CL} = \begin{bmatrix} B_1 + B_2 D_q D_{21} \\ B_q D_{21} \end{bmatrix}$$

$$(12) \quad C_{CL} = [C_1 + D_{12} D_q C_2 \quad D_{12} C_q], D_{CL} = D_{11} + D_{12} D_q D_{21}.$$

Following standard approaches [23, 12], we can parametrize the closed-loop system as:

$$(13) \quad A_{CL} = A_o + \tilde{B} K_o \tilde{C}, B_{CL} = B_o + \tilde{B} K_o \tilde{D}_{21}$$

$$(14) \quad C_{CL} = C_o + \tilde{D}_{12} K_o \tilde{C}, D_{CL} = D_o + \tilde{D}_{12} K_o \tilde{D}_{21}$$

with

$$(15) \quad \tilde{B} = \begin{bmatrix} B_2 \\ 0 \end{bmatrix}, \tilde{C} = \begin{bmatrix} 0 & I \\ C_2 & 0 \end{bmatrix}, \tilde{D}_{12} = D_{12},$$

$$(16) \quad \tilde{D}_{21} = \begin{bmatrix} 0 \\ D_{21} \end{bmatrix}, \text{ and } K_o = [C_q \quad D_q].$$

Thus, the stability of the closed-loop system is determined by $A_o + \tilde{B} K_o \tilde{C}$; which the reader can recognize as the traditional output-feedback form. \square

Substituting the parametrized system into Lemma 1 gives the required condition:

$$(17) \quad F(P, K_o) \triangleq \begin{bmatrix} P & (A_o + \tilde{B}K_o\tilde{C})P & B_o + \tilde{B}K_o\tilde{D}_{21} & 0 \\ P(A_o + \tilde{B}K_o\tilde{C})^T & P & 0 & P(C_o + \tilde{D}_{12}K_o\tilde{C})^T \\ (B_o + \tilde{B}K_o\tilde{D}_{21})^T & 0 & I & (D_o + \tilde{D}_{12}K_o\tilde{D}_{21})^T \\ 0 & (C_o + \tilde{D}_{12}K_o\tilde{C})P & D_o + \tilde{D}_{12}K_o\tilde{D}_{21} & \mu I \end{bmatrix} > 0$$

At this point, traditional approaches would deal with the non-convexity of this condition by lifting the non-convex terms, solving Lemma 1, and extracting K_o from the lifted variable using an inversion. The inversion is problematic when enforcing sparsity constraints and usually researchers require P to be diagonal so that a sparsity pattern can more easily be imposed on the controller. However, there may not be such a P . Here, we take an iterative approach. Define the matrix, F_0 , as:

$$(18) \quad F_0(P, K_o) \triangleq \begin{bmatrix} P & A_oP + \tilde{B}K_o\tilde{C} & B_o + \tilde{B}K_o\tilde{D}_{21} & 0 \\ PA_o^T + (\tilde{B}K_o\tilde{C})^T & P & 0 & PC_o^T + (\tilde{D}_{12}K_o\tilde{C})^T \\ (B_o + \tilde{B}K_o\tilde{D}_{21})^T & 0 & I & (D_o + \tilde{D}_{12}K_o\tilde{D}_{21})^T \\ 0 & C_oP + \tilde{D}_{12}K_o\tilde{C} & D_o + \tilde{D}_{12}K_o\tilde{D}_{21} & \mu I \end{bmatrix} > 0$$

which simply removes P from the non-convex terms. Define the two feasibility programs, \mathbb{P}_0 and \mathbb{P} , as:

$$(19) \quad \mathbb{P}_0(P, K_o) \mapsto \begin{cases} \min_{P, K_o} 0 \\ \text{subject to } Q_i \in S, \forall i \quad \text{and} \\ F_0(P, K_o) > 0 \end{cases}$$

$$(20) \quad \mathbb{P}(P, K_o) \mapsto \begin{cases} \min_{P, K_o} 0 \\ \text{subject to } Q_i \in S, \forall i \\ F(P, K_o) > 0 \end{cases}$$

where $Q_i \in S$ is enforced using equality constraints determined by the zeros in S .

We now define Algorithm 1, to synthesize μ -optimal controllers, using SDP solvers. Below, the notation $\mathbb{P}(P, K_o)^k$ means k -iterations of the feasibility program \mathbb{P} (which is easily specified using the maximum-iterations input of the solver) and the bolded terms identify the decision variables in the program while the other terms are fixed, from the previous iteration.

In practice, Algorithm 1 converged faster when the SDP solver was allowed more iterations when searching for P (i.e. when $k_1 > k_2$). We also found that changing μ affected the speed of convergence. Setting $k = [2, 5, 2]$ and then increasing k_0 with μ set larger than the squared (\mathcal{H}_∞) norm of the closed-loop system with the full controller is a good place to start¹.

2.1. Pattern Discovery. In [14], Lin et al. identified sparsity patterns using “sparsity promoting penalty functions”. Since \mathbb{P}_0 is a relaxation of \mathbb{P} , it can be used to discover sparse patterns by simply adding an objective to \mathbb{P}_0 that minimizes, for example, the re-weighted ℓ_1 -norm of K_o [4].

¹For convenience, full H_∞ controller can be obtained using the MATLAB function `hinfsyn`.

Algorithm 1 Sparse \mathcal{H}_∞ Control Synthesis

Input: $\mu > 0$, FIR order n , sparsity pattern S
Output: (P, K)

```

1:  $K_o \leftarrow \mathbb{P}_0(\mathbf{P}, \mathbf{K}_o)^{k_0}$ 
2: while  $F(P, K_o) \leq 0$  do
3:    $P \leftarrow \mathbb{P}(\mathbf{P}, K_o)^{k_1}$ 
4:   if  $F(P, K_o) > 0$  then
5:     return  $(P, K_o)$ 
6:   else
7:      $K_o \leftarrow \mathbb{P}(P, \mathbf{K}_o)^{k_2}$ 
8:   end if
9: end while

```

Algorithm 2 Pattern Discovery

Input: $\mu > 0$, FIR order n , max iterations N , re-weighting constant $\epsilon > 0$
Output: Pattern(K_o)

```

1:  $W_0 \leftarrow \mathbf{1}$ 
2: while  $k < N$  do
3:    $K_{ok} \leftarrow \begin{cases} \min_{P, K_o} \|W_{k-1} \cdot K_{ok-1}\|_1 \\ \text{subject to } F_0(P, K_o) > 0 \end{cases}$ 
4:    $K_{ok-1} \leftarrow K_{ok}$ 
5:    $W_k(i, j) \leftarrow 1 / (|K_{ok}(i, j) + \epsilon|)$ 
6:    $k \leftarrow k + 1$ 
7: end while
8: return Pattern( $K_o$ )

```

3. EXAMPLES

The examples below were solved using MATLAB/CVX/SeDuMi/SDPT3 [16, 7, 25, 26].

Example 1 This example was used to demonstrate distributed \mathcal{H}_∞ synthesis in [27] and distributed \mathcal{H}_2 synthesis in [9]. The goal is to design a diagonal controller for the (unstable) continuous-time plant below.

$$\begin{aligned}
(21) \quad A &= \begin{bmatrix} -2 & 1 & 1 & 1 \\ 3 & 0 & 0 & 2 \\ -1 & 0 & -2 & -3 \\ -2 & -1 & 2 & -1 \end{bmatrix}, B_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\
B_2 &= \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}, C_1 = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\
C_2 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, D_{11} = 0_3, D_{12} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \\
D_{21} &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, D_{22} = 0_2
\end{aligned}$$

After discretizing using a zero-order hold with a 0.1s sampling period, we redefine A , B_1 , and B_2 as:

$$(22) \quad A = \begin{bmatrix} 0.8189 & 0.08627 & 0.09004 & 0.08133 \\ 0.2524 & 1.003 & 0.03134 & 0.2004 \\ -0.05449 & 0.01017 & 0.7901 & -0.258 \\ -0.1918 & -0.1034 & 0.1602 & 0.8604 \end{bmatrix},$$

$$(23) \quad B_1 = \begin{bmatrix} 0.09531 & 0 & 0 \\ 0.01447 & 0 & 0 \\ 0.08618 & 0 & 0 \\ -0.001083 & 0 & 0 \end{bmatrix}$$

$$(24) \quad B_2 = \begin{bmatrix} 0.004532 & 0.004367 \\ 0.1001 & 0.01005 \\ 0.0003383 & -0.01361 \\ -0.005126 & 0.09363 \end{bmatrix}$$

Running Algorithm 1, using $\mu = 24$ and $k = [10, 5, 2]$, we obtain the following static controller (P is not shown to conserve space):

$$(25) \quad K_0 = \begin{bmatrix} -1.2775 & 0 \\ 0 & -0.5685 \end{bmatrix}$$

which results in $\|T_{zw}\|_\infty = 1.85$ for the continuous-time system. Using the same k , and $\mu = 9$, we obtain the following first-order controller:

$$(26) \quad K_1 = \begin{bmatrix} \frac{-0.6543z-0.5344}{z} & 0 \\ 0 & \frac{-0.1993z-0.2237}{z} \end{bmatrix}$$

which results in $\|T_{zw}\|_\infty = 1.9043$ for the discrete-time system.

The continuous-time equivalent was identified (individually), using the MATLAB function `tfest(id_data,1,0)`, to obtain

$$(27) \quad K_1 = \begin{bmatrix} \frac{-8.469}{s+7.382} & 0 \\ 0 & \frac{-4.167}{s+10.24} \end{bmatrix}$$

which results in $\|T_{zw}\|_\infty = 1.9517$ for the continuous-time system.

$$(28) \quad K_2 = \begin{bmatrix} \frac{-0.6377z^2 - 0.2514z - 0.3369}{z^2} & 0 \\ 0 & \frac{-0.228z^2 - 0.1406z - 0.2332}{z^2} \end{bmatrix}$$

was obtained using $\mu = 9$ and $k = [12, 5, 2]$ – this controller results in $\|T_{zw}\|_\infty = 1.9795$.

The continuous-time equivalent is given by:

$$(29) \quad K_2 = \begin{bmatrix} \frac{-189.7}{s^2 + 40.21s + 157.8} & 0 \\ 0 & \frac{-6.11e04}{s^2 + 565.7s + 1.254e05} \end{bmatrix}$$

and results in $\|T_{zw}\|_\infty = 1.9711$. All three controllers have slightly better closed-loop \mathcal{H}_∞ performance than the ones obtained in [27], the best of which result in $\|T_{zw}\|_\infty = 1.995$.

Example 2 The following continuous-time example is from [20].

$$(30) \quad G(s) = \begin{bmatrix} \frac{1}{s+1} & 0 & 0 & 0 & 0 \\ \frac{1}{s+1} & \frac{1}{s-1} & 0 & 0 & 0 \\ \frac{1}{s+1} & \frac{1}{s-1} & \frac{1}{s+1} & 0 & 0 \\ \frac{1}{s+1} & \frac{1}{s-1} & \frac{1}{s+1} & \frac{1}{s+1} & 0 \\ \frac{1}{s+1} & \frac{1}{s-1} & \frac{1}{s+1} & \frac{1}{s+1} & \frac{1}{s-1} \end{bmatrix}$$

For this model, the following realization is used:

$$(31) \quad A = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$(32) \quad B_1 = \begin{bmatrix} 0.2 & 0 & 0 & 0 & 0 \\ 0 & 0.2 & 0 & 0 & 0 \\ 0 & 0 & 0.2 & 0 & 0 \\ 0 & 0 & 0 & 0.2 & 0 \\ 0 & 0 & 0 & 0 & 0.1 \end{bmatrix}$$

$$(33) \quad B_2 = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$(34) \quad C_1 = \begin{bmatrix} 0.05 & 0 & 0 & 0 & 0 \\ 0.05 & 0.05 & 0 & 0 & 0 \\ 0.05 & 0.05 & 0.05 & 0 & 0 \\ 0.05 & 0.05 & 0.05 & 0.05 & 0 \\ 0.05 & 0.05 & 0.05 & 0.05 & 0.1 \end{bmatrix}$$

$$C_2 = \begin{bmatrix} 0.5 & 0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0.5 & 0 & 0 \\ 0.5 & 0.5 & 0.5 & 0.5 & 0 \\ 0.5 & 0.5 & 0.5 & 0.5 & 1 \end{bmatrix}$$

$$D_{11} = D_{12} = D_{22} = D_{22} = 0_5.$$

After discretizing, using a sampling period of 0.05 s, redefine A , B_1 , and B_2 as:

$$(35) \quad A = \begin{bmatrix} 0.9512 & 0 & 0 & 0 & 0 \\ 0 & 1.051 & 0 & 0 & 0 \\ 0 & 0 & 0.9512 & 0 & 0 \\ 0 & 0 & 0 & 0.9512 & 0 \\ 0 & 0 & 0 & 0 & 1.051 \end{bmatrix}$$

$$(36) \quad B_1 = \begin{bmatrix} 0.009754 & 0 & 0 & 0 & 0 \\ 0 & 0.01025 & 0 & 0 & 0 \\ 0 & 0 & 0.009754 & 0 & 0 \\ 0 & 0 & 0 & 0.009754 & 0 \\ 0 & 0 & 0 & 0 & 0.005127 \end{bmatrix}, B_2 = 10B_1$$

As in [20], we would also like to find a first-order controller that satisfies the sparsity constraint:

$$(37) \quad S = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Setting $\mu = 0.1$ and $k = [2, 5, 2]$, Algorithm 1 obtains the following controller:

$$(38) \quad K_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{-9.265z-5.227}{z} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{-12.68z-15.84}{z} \end{bmatrix}$$

which results in $\|T_{zw}\|_\infty = 0.0163$.

The continuous-time equivalent was identified, using the MATLAB function `tfest(id_data,1,0)`, and re-assembled to obtain:

$$(39) \quad K_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{-120}{s+8.501} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{-616.7}{s+22.48} \end{bmatrix}$$

which results in the same closed-loop cost. Additionally, a static controller can be obtained (using the same settings):

$$(40) \quad K_0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -9.1093 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -12.372 \end{bmatrix}$$

which results in the same cost, $\|T_{zw}\|_\infty = 0.0165$, for both systems.

Example 3 A limitation of our approach is that, presently, SDP solvers can only handle medium-sized problems. For instance, the examples in [14] were too large to solve on a 2-core computer with 4GB of memory. On the other hand, we were able to identify sparser patterns and synthesize sparser controllers than their approach. This example is a smaller version of the mass-spring example in [14]. The plant model is a classical mass-spring system with $N_m = 8$ unit masses and spring constants.

$$(41) \quad A = \begin{bmatrix} 0 & I \\ -T & 0 \end{bmatrix}, \quad B_1 = B_2 = \begin{bmatrix} 0 \\ I \end{bmatrix},$$

$$(42) \quad C_1 = C_2 = Q^{1/2}, \quad D_{11} = D_{21} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

$$(43) \quad D_{12} = D_{22} = \begin{bmatrix} R^{1/2} \\ 0 \end{bmatrix}$$

with T an $N_m \times N_m$ tridiagonal Toeplitz matrix with -2 on the main diagonal and 1 on the sub- and super diagonals. The I and 0 terms are $N_m \times N_m$ identity- and zero matrices, respectively. $R = 10I$ and $Q = I_{2N_m}$. After discretizing, using a sampling period of 0.5 s, the A, B_1 , and B_2 matrices were redefined as in the previous examples.

A controller for this system is partitioned as $K_0 = [K_p, K_v]$ corresponding to the position- and velocity states. Applying Algorithm 2, with $\mu = 80$, $n = 2$, $N = 40$, revealed diagonal patterns for K_p and K_v , which agrees with [14]. However, the entries corresponding to K_p were closer to zero and several times smaller than the K_v terms. Thus, we hypothesized that it is possible to design controllers with the pattern $S = [0, I]$ – this agrees with our intuition that it should be possible to control this system using velocity feedback.

Setting $\mu = 83$ and $k = [10, 5, 2]$, Algorithm 1 obtained the following controller:

$$(44) \quad K_p = 0_{N_m}$$

$$(45) \quad K_v = \text{diag} \left(\begin{bmatrix} -0.7491 \\ -0.706 \\ -0.7255 \\ -0.7316 \\ -0.7316 \\ -0.7255 \\ -0.706 \\ -0.7529 \end{bmatrix} \right)$$

at a cost of $\|T_{zw}\|_\infty = 8.2909$.

4. CONCLUSIONS

We presented a very simple method for solving a very difficult problem that seems to work well for strongly stabilizable plants. Broadly speaking, the idea behind the solution is in line with other approaches to attacking non-convex problems: by breaking it up into smaller problems or steps². The method is aided by the fact that \mathbb{P}_0 tends to provide very good initial guesses and probably deserves more attention on its own.

The method is flexible and, as in Algorithm 2, the feasibility programs can include other objectives. The flexibility can be used to optimize additional objectives and it can be used to enhance convergence for very difficult cases. For example, in Example 3, it is possible to synthesize a still sparser K_v if we minimize $\|P\|_2 + \|Q_i\|_2$ in \mathbb{P}_0 . We left this out for clarity, but it could be researched further.

REFERENCES

- [1] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
- [2] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.
- [3] J. Bu and M. Sznajder. A linear matrix inequality approach to synthesizing low order mixed l_1/h_p controllers. In *American Control Conference, 1997. Proceedings of the 1997*, volume 1, pages 515–519 vol.1, 1997.

²Indeed, the standard LMI approach to robust control synthesis is a great example of how to break-up a non-convex problem into separate convex problems.

- [4] Emmanuel J. Candes, Michael B. Wakin, and Stephen P. Boyd. Enhancing sparsity by reweighted l1 minimization. *stats.ME*, abs/0711.1612, 2007.
- [5] Peng Cheng, Yong-Yan Cao, and Youxian Sun. A new LMI method for strong γ_k - γ_{cl} H_∞ stabilization. In *Control and Automation (ICCA), 2011 9th IEEE International Conference on*, pages 94–99. IEEE, 2011.
- [6] Yung-Shan Chou, Tai-Zu Wu, and Jiunn-Liang Leu. On strong stabilization and H_∞ strong stabilization problems. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 5, pages 5155–5160. IEEE, 2003.
- [7] Inc. CVX Research. CVX: Matlab software for disciplined convex programming, version 2.0. <http://cvxr.com/cvx>, August 2012.
- [8] M.C. de Oliveira, J. Bernussou, and J.C. Geromel. A new discrete-time robust stability condition. *Systems & Control Letters*, 37(4):261 – 265, 1999.
- [9] M.C. de Oliveira and J.C. Geromel. Design of decentralized dynamic output feedback controllers via a separation principle. *International Journal of Control*, 73(5):371–381, 2000.
- [10] M.C. de Oliveira, J.C. Geromel, and J. Bernussou. Extended h_2 and h_∞ norm characterizations and controller parametrizations for discrete-time systems. *Int. J. Control*, 75(9):666–679, 2002.
- [11] Geir E. Dullerud and Fernando Paganini. *A Course in Robust Control Theory: A Convex Approach*. Springer, New York, NY, USA, 2000.
- [12] Pascal Gahinet and Pierre Apkarian. A linear matrix inequality approach to h_∞ control. *International Journal of Robust and Nonlinear Control*, 4:421–448, 1994.
- [13] Jean-Bernard Lasserre. *Moments, positive polynomials and their applications*, volume 1. World Scientific, 2009.
- [14] Fu Lin, M. Fardad, and M.R. Jovanovic. Design of optimal sparse feedback gains via the alternating direction method of multipliers. *Automatic Control, IEEE Transactions on*, 58(9):2426–2431, Sept 2013.
- [15] J. Lopez, Y. Wang, and M. Sznaier. Sparse h_2 optimal filter design via convex optimization. In *American Control Conference (ACC), 2014*, pages 1108–1113, June 2014.
- [16] MATLAB. *version 8.3.0.532 (R2014a)*. The MathWorks Inc., Natick, Massachusetts, 2014.
- [17] F. Palacios-Quinonero, J. Rubio-Massegu, J.M. Rossell, and H.R. Karimi. Discrete-time static output-feedback semi-decentralized h_∞ controller design: An application to structural vibration control. In *American Control Conference (ACC), 2012*, pages 6126–6131, June 2012.
- [18] Ian R Petersen. Robust control of an uncertain system via a stable output feedback controller. *Automatic Control, IEEE Transactions on*, 54(6):1418–1423, 2009.
- [19] B. Polyak, M. Khlebnikov, and P. Shcherbakov. An lmi approach to structured sparse feedback design in linear control systems. In *Control Conference (ECC), 2013 European*, pages 833–838, July 2013.
- [20] Michael Rotkowitz and Sanjay Lall. A characterization of convex problems in decentralized control*. *Automatic Control, IEEE Transactions on*, 51(2):274–286, 2006.
- [21] H. Rotstein and A. Sideris. h_∞ optimization with time-domain constraints. *Automatic Control, IEEE Transactions on*, 39(4):762–779, 1994.
- [22] Hector Rotstein and Athanasios Sideris. h_∞ optimal control with time-domain constraints: numerical aspects and examples. In *American Control Conference, 1993*, pages 2767–2771, 1993.
- [23] Ricardo S. Sanchez-Pena and Mario Sznaier. *Robust Systems Theory and Applications*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1998.
- [24] B.L. Stevens and F.L. Lewis. *Aircraft Control and Simulation*. Wiley, 2003.
- [25] J.F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11–12:625–653, 1999. Version 1.05 available from <http://fewcal.kub.nl/sturm>.
- [26] K. C. Toh, M.J. Todd, and R. H. Tutuncu. Sdpt3 – a matlab software package for semidefinite programming. *Optimization Methods and Software*, 11:545–581, 1999.
- [27] R.J. Veillette, J.B. Medanic, and William R. Perkins. Design of reliable control systems. *Automatic Control, IEEE Transactions on*, 37(3):290–304, Mar 1992.
- [28] Y. Wang, J. Lopez, and M. Sznaier. Sparse static output feedback controller design via convex optimization. In *IEEE CDC 2014*. IEEE, December 2014.
- [29] K. Zhou and J.C. Doyle. *Essentials of Robust Control*. Prentice Hall Modular Series for Eng. Prentice Hall, 1998.